DETAILED ACTION

This Office action is in response to the Applicant's response filed on 05/30/08.

Status of Claims

Claims 1 – 11, 13 – 26, & 28 – 49 are pending in the Application.

Claims 1, 11, 20 – 22, 32, 38, & 48 have been amended.

Claims 12 & 27 are cancelled.

Claims 1 – 11, 13 – 26, & 28 – 49 are rejected.

Telephone Interview

The Examiner and the Applicant held a telephone interview on Friday, January 16, 2009. The Examiner thanks the Applicant for his courtesy and his work to bring this application to allowance.

Response to Amendment

Applicant's amendments and arguments filed on 09/30/08 in response to the Office action mailed on 05/30/08 have been fully considered, but they are not persuasive. Therefore, the rejections made in the previous Office action are maintained, and restated below, with changes as needed to address the amendments.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 11, 13-19 and 49 are rejected under 35 U.S.C. 103(a) as being unpatentable over ***Hitz*** et al. (US Patent 5,819,292), hereinafter ***Hitz***, in view of ***Bush*** et al. (US Patent 5,790,778), hereinafter ***Bush***.

In regard to claim 11, Hitz teaches a method for detecting leaked buffer writes between a first consistency point and a second consistency point, comprising:

selecting a data buffer (a buffer is selected to store the inode which contains the meta-data file – col. 9, lines 19-49. The meta-data file comprises, *inter alia*, a block map – col. 5, lines 48-59. Hitz further describes the block map as consisting of information for up to 20 snapshots. Since the consistency points are classified as points after a snapshot, the block map comprises the information needed to ascertain the consistency point number – col. 4, lines 6-43);

determining if the selected data buffer includes a buffer check control structure (the system inherently makes a determination if the buffer check control structure within an inode is present simply by referring to the data recorded within it);

determining, in response to the selected data buffer including a buffer check control structure, if a consistency point number within the buffer check control structure is correct, and if so, performing a write operation of the file system buffer (before

converting to a new consistency point, the system will perform a check sum on the fsinfo structure (containing the root inode which comprises the consistency point information) to determine if the one of the copies has been corrupted in some way. Once consistency is determined, the system will continue to write the new node to the disk – col.12, lines 11-48).

Though the Hitz discloses storing consistency point numbers in the buffer check control structure, he fails to further teach determining if one or more uniquely identifying numbers (hereinafter magic numbers) are within the data buffer check control structure, wherein the magic numbers are used to uniquely identify the raw data buffer and to indicate that the data buffer needs to be checked for leakage as recited in the instant claim.

Bush however teaches a system for simulated program execution error detection, which utilizes a data chunk (Fig. 17, element 1700), comprises a plurality of entries (i.e. collectively the magic number) which include a flag for determining if a memory buffer needs to be checked for leakage (e.g., reachable flag – element 1702), a chunk number, and a plurality of pointers used to uniquely identify buffers and control structures) – col. 28, l. 53 through col. 29, l. 28.

It would have been obvious to one of ordinary skill in the art at the time of the invention for Hitz to further include Bush's system for simulated program execution error detection into his own system for maintaining consistency states in a file system. By doing so, Hitz would have a means of more quickly identifying, and correcting, programming errors that may occur in his system as taught by Bush in col. 3, ll. 1-17.

As for claims 13 and 14, Hitz fails to teach magic numbers as recited in this claim. Bush however teaches magic numbers (see col. 28, l. 53 through col. 29, l. 28), and even though he

does not explicitly teach the magic number as comprising either a 64-bit number, two 32-bit number, nor the consistency point number as comprising a 32-bit number, such limitations are merely a matter of design choice and would have been obvious in the system of Bush. These limitations fail to define a patentably distinct invention over Bush since both the invention as a whole and that of Bush are directed to storing a magic number used to uniquely identify the data block; and storing a consistency number, used to track certain points in time the system maintained a consistent state.

As for claim 49, though Hitz teaches checking consistency points (as per the rejection of claim 11, above), he fails to explicitly teach detecting buffer leakage via the aid of magic numbers, when the magic number for the buffer check control structure is correct, but the consistency point number is not.

Bush however teaches a system for simulated program execution error detection, which utilizes a data chunk (Fig. 17, element 1700), comprises a plurality of entries (i.e. collectively the magic number) which include a flag for determining if a memory buffer needs to be checked for leakage (e.g., reachable flag – element 1702), a chunk number, and a plurality of pointers used to uniquely identify buffers and control structures) – col. 28, l. 53 through col. 29, l. 28. It is worthy to note that by Applicant's own disclosure, a memory leak occurs when a consistency point is not correct. In other words, the combined teachings of Hitz and Bush ensure that's Bush's reachable flag is used to determine if a consistency point is correct or not.

It would have been obvious to one of ordinary skill in the art at the time of the invention for Hitz to further include Bush's system for simulated program execution error detection into his own system for maintaining consistency states in a file system. By doing so, Hitz would

have a means of more quickly identifying, and correcting, programming errors that may occur in his system as taught by Bush in col. 3, ll. 1-17.

As for claim 15, Hitz teaches the method of claim 11 wherein the step of determining if the consistency point number is correct further comprises the step of determining if the consistency point number within the buffer check control structure equals a consistency point number identifying a current consistency point (col. 11, line 62 through col. 12, line 38 – the system maintains two identical copies of the root inode containing the information of the consistency point of the system. The system can then compare the current root inode with the copy to determine if the consistency point is accurate, and that no failure has occurred).

As for claim 16, Hitz teaches the method of claim 11 wherein the step of performing a write operation further comprises a step of writing a set of raw data within the data buffer to disk (all data written is written to the disk – see abstract. The data is buffered before flushed and written to the disk).

As for claim 17, Hitz teaches the method of claim 16 wherein the raw data comprises the buffer check control structure (both the buffer data structure (1010B), and the pointers (1010C) are stored and associated with the inode, hence comprise the raw data buffer associated with the data buffer – col. 7, lines 5-41).

As for claim 18, Hitz teaches the method of claim 16 wherein the step of performing the write operation further comprises a step of removing the buffer check control structure from the data before writing the file system buffer to disk (the buffered data is flushed (i.e. removed) from the buffer before it is written to the newly allocated regions on the disk - col. 12, lines 9-24).

As for claim 19, Hitz teaches the method of claim 16 wherein the step of performing the write operation comprises the step of writing only the raw data within the file system buffer to disk (col. 12, lines 9-24 – the raw data is the only data flushed to the disk during the update).

Claims 1-10, 20-26 and 28-48 are rejected under 35 U.S.C. 103(a) as being unpatentable over ***Hitz*** (US Patent 5,819,292), in view ***Marion*** et al. (US PG Publication 2003/0163661 A1), hereinafter ***Marion***, and in further view of ***Bush*** (US Patent 5,790,7781).

In regard to claims 1, 20 and 22, Hitz teaches method for detecting leaked buffer writes between a first consistency point and a second consistency point in a data storage system, the method comprising:

receiving a write operation, the write operation identifying a file for the write operation (Hitz's invention is directed to managing changes to a file system. In his disclosure, Hitz describes a Write Anywhere File-System Layout (WAFL) directed to writing new data (i.e. files) to the file system (i.e. storage system of claim 22). The files can write new inode files to the file system – col. 4, lines 6-32);

creating a data buffer associated with the write operation (referring to Fig. 10, when a new incore inode is created (1010A), an area is allocated to the inode in order to store information, a WAFL buffer structure (1010B), a set of pointers (1010C), and an on-disk inode (1010D). The pointers point to the newly created indirect WAFL buffers (1020) – col. 7, lines 5-41). Note the area needed to store these elements is created (i.e. allocated) as Hitz explicitly describes his invention as writing new data to unallocated blocks on a disk – see abstract; and

Though Hitz teaches writing a buffer check control structure to a raw data buffer associated with the data buffer (again both the buffer data structure (1010B), and the pointers (1010C) are stored and associated with the inode, hence comprise the raw data buffer associated with the data buffer – col. 7, lines 5-41), he fails to do so in response to determining the volume has buffer leakage detection activated as recited in these claims.

Marion however teaches a system and method for tracking memory leaks which checks a memory leak flag to track memory allocation and de-allocation activities (paragraphs 0010, and 0011, all lines). More specifically, (referring to Fig. 3, elements 310, 320 and 330 and paragraph 0046-0048, all lines), a determination is made to track memory leaks or not (leak detection activation). Based on this determination, the remaining memory allocation executes. Note by combining the teachings of Hitz and Marion, they would be able to check for memory leaks prior to allocating new files, and subsequently prior to writing the buffer control check structure to the raw buffer (i.e. in response to the determination).

Though the Hitz discloses storing consistency point numbers in the buffer check control structure, he fails to further teach determining if one or more uniquely identifying numbers (hereinafter magic numbers) are within the data buffer check control structure, wherein the magic numbers are used to uniquely identify the raw data buffer and to indicate that the data buffer needs to be checked for leakage as recited in the instant claim.

Bush however teaches a system for simulated program execution error detection, which utilizes a data chunk (Fig. 17, element 1700), comprises a plurality of entries (i.e. collectively the magic number) which include a flag for determining if a memory buffer needs to be checked for

leakage (e.g., reachable flag – element 1702), a chunk number, and a plurality of pointers used to uniquely identify buffers and control structures) – col. 28, l. 53 through col. 29, l. 28.

It would have been obvious to one of ordinary skill in the art at the time of the invention for Hitz to further include Marion's method of tracking memory leaks into his own system for maintaining consistency states of a file system during file allocation. By doing so, Hitz could benefit by having an efficient means of memory leak detection, which in turn could help improve system performance as taught by Marion, paragraphs 0007-009, all lines.

It would have been obvious to one of ordinary skill in the art at the time of the invention for Hitz to further include Bush's system for simulated program execution error detection into his own system for maintaining consistency states in a file system. By doing so, Hitz would have a means of more quickly identifying, and correcting, programming errors that may occur in his system as taught by Bush in col. 3, ll. 1-17.

As for claim 6, though the combined teachings of Hitz and Marion disclose storing consistency point numbers in the buffer check control structure, they fail to further teach storing one or more magic numbers in which the magic numbers uniquely identify a particular buffer check control structure.

Bush however teaches a system for simulated program execution error detection, which utilizes a data chunk (Fig. 17, element 1700), comprises a plurality of entries (i.e. collectively the magic number) which include a flag for determining if a memory buffer needs to be checked for leakage (e.g., reachable flag – element 1702), a chunk number, and a plurality of pointers used to uniquely identify buffers and control structures) – col. 28, l. 53 through col. 29, l. 28.

It would have been obvious to one of ordinary skill in the art at the time of the invention for Hitz to further include Bush's system for simulated program execution error detection into his own system for maintaining consistency states in a file system. By doing so, Hitz would have a means of more quickly identifying, and correcting, programming errors that may occur in his system as taught by Bush in col. 3, ll. 1-17.

As for claim 21, though the combined teachings of Hitz, Marion and Bush disclose all the limitations of the claim (as per the rejection of claims 1, *supra*), they fail to specifically teach said method as being implemented on a computer readable media with instructions executed by a processor as recited in this claim.

Examiner takes Official Notice (see MPEP § 2144.03) that implementing a method to be performed on a file system (as described by Hitz) via the use of a "computer readable media containing instructions for execution on a processor" in a computing environment was well known in the art at the time the invention was made.

One of ordinary skill in the art would have been motivated to implement the method as instructions on a computer readable media, as it is well known in the art that such implementation is inexpensive, and easy to maintain.

Since Applicant failed to traverse the examiner's previous assertion of Official Notice, the use of a "computer readable media containing instructions for execution on a processor" is taken to be admitted prior art pursuant to MPEP § 2144.03, subsection C.

As for claims 7, 8 and 10, though neither Hitz nor Marion teach magic numbers, Bush does in fact teach magic numbers (as per claim 1 above), even though he does not explicitly teach the magic number as comprising either a 64-bit numbers, two 32-bit numbers, nor the

consistency point number as comprising a 32-bit number. Such limitations however they are merely a matter of design choice and would have been obvious in the system of Bush. These limitations fail to define a patentably distinct invention over Bush since both the invention as a whole and that of Bush are directed to storing a magic number used to uniquely identify the data block; and storing a consistency number, used to track certain points in time the system maintained a consistent state.

As for claims 2 and 23, Hitz teaches creating the data buffer as further comprising the step of creating a buffer control structure and a raw data buffer (the structures previously described in the rejection of claim 1 illustrate the buffer control structure and raw data buffer which are created when the incore inode is created (i.e. allocated)).

As for claims 3 and 24, Hitz teaches the buffer control structure as comprising a pointer to the raw data buffer (the buffer structure contains pointers to reference the 16 buffer pointers (1010C) - col. 7, lines 5-16).

As for claim 4, Hitz teaches the method of claim 1 wherein the step of writing the buffer check control structure to the raw data buffer further comprises the steps of:

creating the buffer check control structure (again, the buffer check control structure is created upon allocation of the inode); and

overwriting a portion of the raw data buffer with the buffer check control structure (a portion of the raw data buffer is comprised of the buffer structure (1010B)).

As for claims 5 and 26, Hitz teaches the step of writing the buffer check control structure to the raw data buffer as further comprising the steps of:

creating the buffer check control structure (again, the buffer check control structure is created upon allocation of the inode); and

associating the buffer check control structure to the raw data buffer in a contiguous block of memory (the buffer check is associated with the raw data buffer as it contains pointers that reference a block within the buffer itself – col. 7, lines 5-41. Note by definition a block of data is contiguous; therefore the information is inherently stored contiguously. Additionally, Fig. 10 depicts the information as being stored contiguously in blocks).

As for claim 9, Hitz teaches his consistency point number as identifying a current consistency point (again, the most recently recorded consistency point is indicative of the system's most current point of consistency).

As for claim 25, Hitz teaches overwriting the buffer (col. 23, lines 45-67 – see also Figs. 23A and 23B).

Claim 38 is similar in scope to claims 1 and 22, therefore it is rejected based on the same rationale as those claims discussed *supra*.

Claim 48 is similar in scope to claim 38; therefore it is rejected based on the same rationale as claim 38 *supra*.

Claim 32 is rejected based on the same rationale as claim 6 as discussed above.

Claims 28-31 are rejected based on the same rationale as claims 7-10 respectively.

Claims 33, 34, 35, 36 and 37 contain similar limitations as claims 1, 9, 2, 4 and 5 respectively; therefore they are rejected based on the same rationale as these claims.

Claims 39-47 contain similar limitations as claims 2-10 respectively; therefore they are rejected based on the same rationale as these claims.

Response to Arguments

Applicant's arguments filed 09/30/08 have been fully considered but they are not persuasive.

Applicant argues *the combination of Hitz and Bush teaches away from the claimed invention in that Bush executes a program over and over again checking for errors after the program has executed many times*. Examiner disagrees. Both applications are preventing errors in programs that are running. This makes them analogous to the instant application. Furthermore, it does not matter how many times a program runs in order to check for memory leaks because this is not in the scope of the claim language.

Applicant argues *a person of ordinary skill in the art would be led astray from Applicant's claimed invention by following the teachings of Marion*. Setting flags is well known in the art. The flags in Marion are used merely to determine whether or not to check for memory leaks. The claim language states determining that buffer leakage detection is active. Checking the value of a flag that states whether memory leaks are checked would indicate whether buffer leakage protection is activated.

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Examiner's Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Shawn Eland whose telephone number is (571) 270-1029. The examiner can normally be reached on MO - TH, & every other FR.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Hyung Sough can be reached on (571) 272-6799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Hyung S. Sough/
Supervisory Patent Examiner, Art Unit 2188
01/21/09

/Shawn Eland/
Examiner, Art Unit 2188
1/26/2009